Potential Issues and Mitigation in Migrating Embedded Systems to Multicore

Redge Bartholomew

Rockwell Collins

Report Documentation Page

Form Approved
OMB No. 0704-0188

Public reporting burden for the collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to a penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number.

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED
APR 2010		**00-00-2010 to 00-00-2010**

4. TITLE AND SUBTITLE	5a. CONTRACT NUMBER
Potential Issues and Mitigation in Migrating Embedded Systems to Multicore	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **Rockwell Collins,400 Collins Road N.E.,Cedar Rapids,IA,52498**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES
Presented at the 22nd Systems and Software Technology Conference (SSTC), 26-29 April 2010, Salt Lake City, UT. Sponsored in part by the USAF. U.S. Government or Federal Rights License

14. ABSTRACT

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**	**Same as Report (SAR)**	**12**	

Embedded System Migration To Multicore

- Over time, as systems absorb capability, software grows & demand for processor capacity increases
  - Same holds as systems & subsystems are consolidated onto common computing platforms
- In the past, CPU/clock acceleration met need
- Currently, further CPU acceleration requires prohibitively expensive increase in power, heat/cooling, surface area
- As a result, performance improvement requires adding CPUs (cores)
  - Number of cores per chip now doubles every 18-24 months

The Embedded Software Problem

- Where a single core will always sustain a system's software, multicore migration will have low impact
- Where clock speed is reduced to cut power, size, cost, …
- … or system absorbs features that consume available processor capacity, …
- … it becomes increasingly less likely that a single CPU will support all of a system's software
  - It can no longer run as a sequential monolith and must be split into multiple tasks running in parallel on separate cores

Sequential systems with high cohesion may produce tightly coupled & highly problematic parallel tasks

Multicore Impact – Software Parallelism

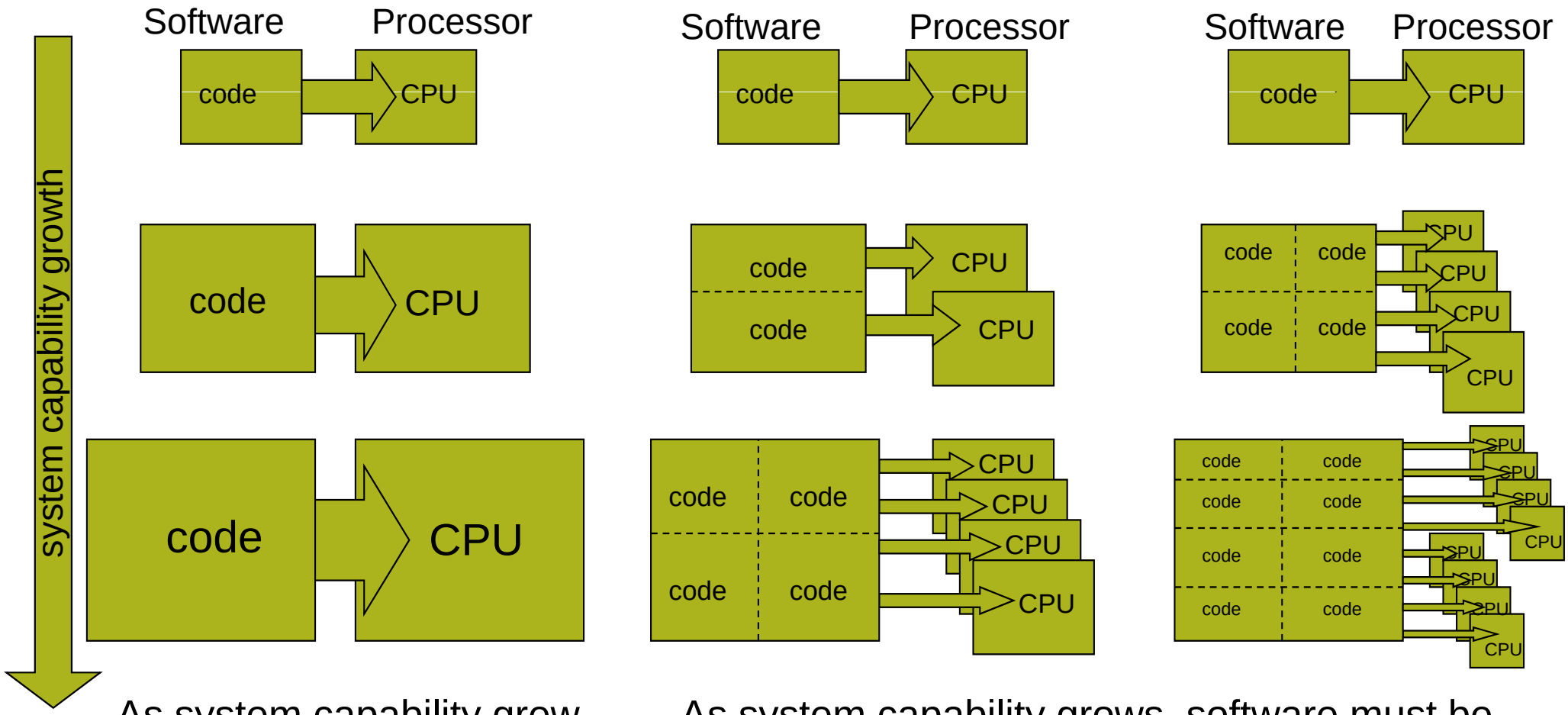
Single CPU (past)

Increase CPU Speed with growth
(increase CPU power, heat)

Multicore

Add CPUs with growth
(maintain CPU speed, power, heat)

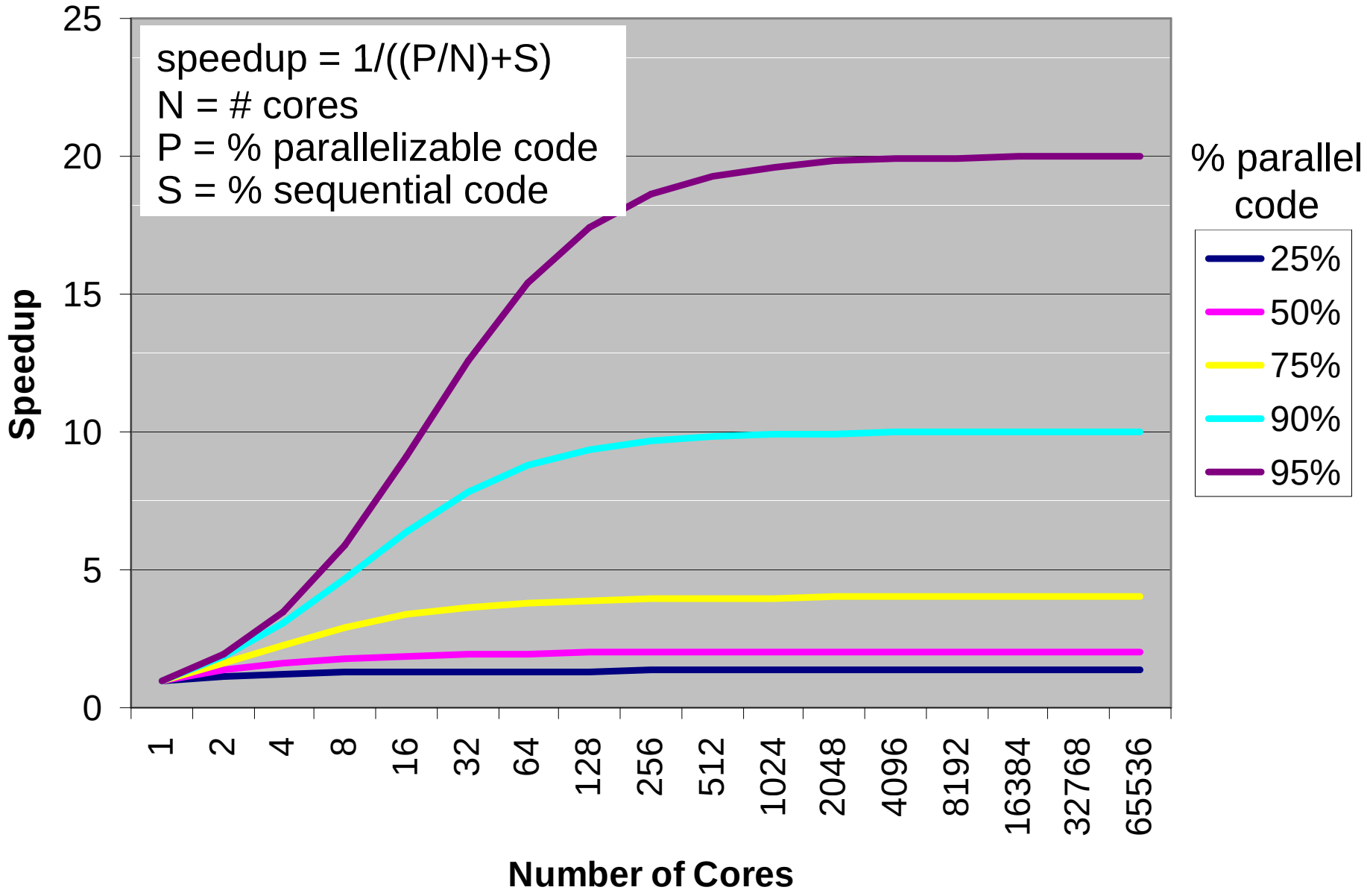
Add slower CPUs with growth
(SWAP constrained)
(reduce CPU size, power, heat)



As system capability grew, CPU speed grew with it

As system capability grows, software must be split into parallel tasks running on separate CPUs

Amdahl's Limit: Adding Cores Helps Only If Software Can Run in Parallel



Software Parallelism

- Application parallelism – independent applications run in parallel on separate cores
- Data parallelism – loops, repetitive manipulations of large datasets: identical code is vectored to multiple cores
  - e.g., driving segments of a display, matrix arithmetic
- Task parallelism – time-driven or event-driven tasks executed in parallel on separate cores
  - e.g., sequential math, complex logic, processing independent I/O ports, resource/sensor management

Much of embedded software is susceptible only to task parallelism

Task-Parallel Software Requires More Effort

- Find potential parallelism in architectures & designs
- Enforce correct execution sequence among tasks
- Assign tasks to cores
- Balance processing/communication load among cores,
- Manage message passing & shared memory access
- Determine/implement failure recovery mechanism
  - As cores added & scale approaches nanometer level, hardware runtime failure rate greatly increases
  - Redundancy may not be cost effective in meeting reliability spec

Task-Parallel Software Requires More Effort

- Integrate a multitasking system across multiple cores
  - Suspend/resume tasks, maintain required task execution sequence
- Optimize, debug, verify code across multiple cores
- Find, resolve race, deadlock, livelock
- Find, eliminate performance bottlenecks
- Minimize parallelism overhead (e.g., context switching)
- Optimize main memory/bus access among cores
- Etc. …

Software Development Problem

- Available tools typically apply to data parallelism and server applications
- Tools for developing task-parallelism are for the most part still emerging – development is largely manual
  - Available tools are limited in scope/application & effectiveness, or require significant annotations in the source code
- Result is that mapping sequential code onto embedded multicore chips is time-consuming and error-prone
  - It may produce highly coupled tasks with significant problems in task-sequencing, shared-data access, and message passing

At least initially, software development productivity could decline significantly

Mitigation - Training

- Typically, embedded software engineers have little experience developing parallel software
  - Few subsystems in the past required multiple processors, and many of those involved independent functions with little coupling
- Training needs
  - Architecting, designing, implementing, verifying parallel software
  - Developing multitasking systems & using a multi-tasking RTOS
  - Developing shared memory systems, detecting race, deadlock, livelock, message passing, load balancing
  - Designing task synchronization, time-correlating test data
  - Developing lightweight autonomic mechanism to meet reliability spec – e.g., without prohibitively expensive redundancy

Mitigation – Migration Planning

- Assume CPUs will get slower as more are added to a single chip
- Move single processor software to single core of target chip & determine change in performance
  - Plan accordingly for near-term, long-term
- Move software monoliths to singe core, where possible, but plan for deconstructing it into parallel tasks
- Where available, move those with multiprocessor software development experience to multicore migration team
- Exploit research, prototypes at I/UCRCs, UARCs, FFRDCs

Acronyms

- CPU – Central Processing Unit
- SWAP – Size, Weight, and Power
- I/O – Input & Output
- RTOS – Real-Time Operating System
- I/UCRC – Industry/University Cooperative Research Center
- UARC – University-Affiliated Research Center
- FFRDC – Federally Funded Research/Development Center